Solving Multi Constrained Team Orienteering Problems to Generate Tourist Routes

Ander Garcia[a,*], Pieter Vansteenwegen[b], Wouter Souffriau[b,d], Olatz Arbelaitz[c], Maria Teresa Linaza[a]

[a] *Department of eTourism and Cultural Heritage, Visual Communication Technologies VICOMTech, Spain*
[b] *Centre for Industrial Management, Katholieke Universiteit Leuven, Belgium*
[c] *Department of Computer Architecture and Technology, University of the Basque Country, Spain*
[d] *KaHo St Lieven Gebr Desmetstraat 1, 9000 Gent, Belgium*

Abstract

Generating personalized tourist routes based on a tourist's interests and constraints is an upcoming trend in tourist applications. This problem can be directly related to the Multi Constrained Team Orienteering Problem with Time Windows (MCTOPTW). The MCTOPTW consists of a set of locations, each of them with a certain score, a time window and one or more associated attributes, such as an entrance fee. Each attribute type has an associated constraint with a maximum allowed value for a route, such as a limited budget. Visiting a location within its time window allows collecting its score as a reward. The goal is to determine a fixed number of routes that maximize the collected score without violating any of the constraints. This paper describes an iterated local search metaheuristic to solve the MCTOPTW, in order to generate personalized tourist routes in real-time. Extensive experimental results with new and existing benchmark instances prove the appropriateness of the algorithm.

Key words: Multi Constrained Team Orienteering Problem with Time Windows, Metaheuristics, Iterated Local Search, Tourist routes
2010 MSC: 90C59 Approximation methods and heuristics, 90B06 Transportation, logistics, 90C90 Applications of mathematical programming

1. Introduction

Nowadays, the creation of personalized tourist routes that take into account the profile of the tourist and up-to-date attraction information is a time consuming task. This task is often done by the staff of the Local Tourist Organizations (LTOs).

When tourists are at a destination and search for information in the LTO, the staff categorizes their profile (cultural, romantic, family, etc.) and restrictions (time, money, etc.). Combining this

*Corresponding author: Paseo Mikeletegi 57, E-20009 Donostia - San Sebastian, Spain, Tel: +[34]943309230, Fax: +[34]943309393

Email addresses: agarcia@vicomtech.org (Ander Garcia), pieter.vansteenwegen@cib.kuleuven.be (Pieter Vansteenwegen), wouter.souffriau@cib.kuleuven.be (Wouter Souffriau), wouter.souffriau@kahosl.be (Wouter Souffriau), olatz.arbelaitz@ehu.es (Olatz Arbelaitz), mtlinaza@vicomtech.org (Maria Teresa Linaza)

information with their knowledge about local attractions (location, price, timetable, etc.), they suggest a personalized route for the tourist.

This route does not take into account new circumstances that may happen during the visit, as spending more time than expected at an attraction. In such cases, tourists have to go back to the LTO or ask local citizens in order to update their route properly. Otherwise they have to follow the more obvious tourist paths.

A Personalized Electronic Tourist Guide (PET) should perform at least the same task fulfilled by the LTO, on a hand-held device. A PET should create routes maximizing tourists' satisfaction taking into account different restrictions, attractions' attributes (opening and closing times, visit duration, entrance fee, etc.) and traveling times. A PET should provide an integrated solution for route planning that adapts to new circumstances in real time: tourists do not expect to wait minutes to receive a new route. Vansteenwegen [1] presented an extensive review of existing PETs and none of these guides applied advanced heuristics to solve the route creation problem. PETs have also been called Mobile Tourist Guide (MTG), Personal Navigation Systems for tourism (PNS) and Electronic Tourist Guides (ETG).

The requirements of the problem a PET has to meet can be mapped into the Tourist Trip Design Problem (TTDP) [2, 3, 4]. The Orienteering Problem (OP) and its extensions, including the Multi Constrained Team Orienteering Problem with Time Windows (MCTOPTW), are simplified versions of the TTDP. Zenker and Ludwig [5] categorize route planning problems according to their orientation to single or multiple destinations. They indicate that the Team Orienteering Problem with Time Windows (TOPTW) can be applied to problems with multiple destinations and time windows. When constrained attributes are included, they do not find any alternative model than the MCTOPTW.

The application we are working on is called "Citytrip Planner" and it is available online since July 2009 (http://www.citytripplanner.com). It proposes a personalized city trip to a tourist, based on a limited number of inputs, such as the number of days of the visit or the starting and ending time and points for each day. Solving the MCTOPTW in real time will allow us to extend the website functionality, adapting the output route to tourists' restrictions and making the trip planning available on mobile devices.

We are the first to develop an algorithm that solves these difficult and relevant MCTOPTW instances with limited computational efforts. The algorithm is based on an existing algorithm for the TOPTW. We have extended and adapted it to solve the MCTOPTW. Furthermore, we have compared its results against existing solutions for a similar problem (Selective Vehicle Routing Problem with Time Windows, SVRPTW) and we have proposed new test sets to allow researchers to compare new heuristics with our algorithm.

This paper is organized as follows. In the next Section we review existing literature. In Section 3 we give a mathematical problem definition. In Section 4, we describe the metaheuristic, focusing on the changes with the algorithm for the TOPTW. In Section 5, extensive experimental results are presented. Conclusions and further work are discussed in Section 6.

2. State of the Art

In the OP [6], several locations with an associated score have to be visited in order to obtain a total trip score. A player can visit each location only once. The objective is to obtain a total trip score as high as possible without violating a time restriction. Its generalization to multiple players in a team is known as the Team Orienteering Problem (TOP). Vansteenwegen et al. [7] present an extensive review of these problems, solution algorithms and applications.

When locations have an associated time window, the problem is called the TOPTW [8]. Righini and Salani [9] applied bi-directional dynamic programming to solve the Orienteering Problem with Time Windows (OPTW) instances to optimality. Regarding the TOPTW, Tricoire et al. [10] presented a variable neighborhood search algorithm for a generalization of the TOPTW called the Multi Period OP with Multiple Time Windows (MuPOPTW) . Vansteenwegen et al. [11] developed an efficient metaheuristic to solve the TOPTW. Montemanni and Gambardella [12] proposed using ant colony systems for the TOPTW. Interested readers can find a thorough revision of literature about the (T)OPTW in these four papers.

The MCTOPTW extends the TOPTW by adding multiple constraints, such as a limited money budget. None of the above mentioned TOPTW algorithms can tackle problems with multiple constraints. To the best of the authors' knowledge, no publication about the MCTOPTW is available. However, related problems occur in the literature. Archetti et al. [13] defined a Capacited TOP as a special case of the TOP, where each location has a certain demand and each vehicle a certain capacity that cannot be exceeded. This can be seen as a special case of the MCTOPTW with only one extra constraint and no time windows.

The SVRPTW was introduced by Gueguen [14] and Hayari et al. [15]. In the SVRPTW not all locations can be visited because i) the length of routes is limited due to a distance constraint, ii) the time duration of the routes is limited, iii) vehicles have a restricted capacity to serve locations, who have a certain demand associated. The objective of the SVRPTW is to obtain the maximum possible reward not violating any of its constraints. The difference between the SVRPTW and the MCTOPTW is the constraint type of the distance constraint. The attribute constraints in the MCTOPTW are based on attributes linked to each individual location (entrance fee, visiting duration, etc.), while the distance constraint in the SVRPTW is based on distances between different locations and therefore dependent of the sequence of visits. Moreover, the MCTOPTW can have more attribute constraints than the SVRPTW. Boussier et al. [16] designed an exact solution approach, based on a branch & price algorithm, to solve the SVRPTW.

Although the MCTOPTW is a relevant problem, for instance to design personalized tourist routes, no algorithms have been developed to deal with this problem (in real-time).

3. Problem definition

In the MCTOPTW, a set of N locations i is given, each with a score S_i, a service or visiting time T_i, a time window for the start of the visit $[O_i, C_i]$ and attributes e_{ik}, related to K different attribute constraints. Value e_{ik} can e.g. be a capacity or an entrance fee of location i. S_i denotes the score of location i for a given tourist, in other words: "how much a given tourist is interested in location i".

The starting location 1 and the end location N of every route are fixed. The time c_{ij} needed to travel from location i to j is known for all locations. It might be impossible to visit all locations since the available time is limited to a given time budget T_{max} and each attribute constraint k has an upper bound E_k.

The goal of the MCTOPTW is to determine M routes, each limited by T_{max} and E_k, that visit some of the locations during their appropriate time window, and that maximize the total collected score. Each location can be visited at most once.

With these notations, the MCTOPTW can be formulated as an integer problem.

Numerical data:
S_i = the score associated to location i

e_{ik} = the value related to attribute constraint k associated to location i
E_k = upper bound of attribute constraint k of each route
c_{ij} = travel time from location i to location j
T_i = time duration of a visit to location i
T_{max} = time budget of each route
$[O_i, C_i]$ = time window of location i
M = number of routes
N = number of locations
K = number of attribute constraints
P = a large constant

Decision variables:
$x_{ijd} = 1$ if in route d, a visit to location i is followed by a visit to location j, 0 otherwise
$y_{id} = 1$ if location i is visited in route d, 0 otherwise
s_{id} = the start of the service at location i in route d

$$\max \sum_{d=1}^{M} \sum_{i=2}^{N-1} S_i y_{id} \tag{1}$$

$$\sum_{d=1}^{M} \sum_{j=2}^{N} x_{1jd} = \sum_{d=1}^{M} \sum_{i=1}^{N-1} x_{iNd} = M \tag{2}$$

$$\sum_{i=1}^{N-1} x_{iod} = \sum_{j=2}^{N} x_{ojd} = y_{od} (o = 2, ..., N-1; d = 1, ..., M) \tag{3}$$

$$s_{id} + T_i + c_{ij} - s_{jd} \leq P(1 - x_{ijd}) (i, j = 1, ..., N; d = 1, ..., M) \tag{4}$$

$$\sum_{d=1}^{M} y_{id} \leq 1 (i = 2, ..., N-1) \tag{5}$$

$$O_i \leq s_{id} (i = 1, ..., N; d = 1, ..., M) \tag{6}$$

$$s_{id} \leq C_i (i = 1, ..., N; d = 1, ..., M) \tag{7}$$

$$\sum_{i=1}^{N-1} (T_i y_{id} + \sum_{j=2}^{N} c_{ij} x_{ijd}) \leq T_{max} (d = 1, ..., M) \tag{8}$$

$$\sum_{i=1}^{N} e_{ik} y_{id} \leq E_k (k = 1, ..., K; d = 1, ..., M) \tag{9}$$

$$x_{ijd}, y_{id} \in 0, 1 (i, j = 1, ..., N; d = 1, ..., M) \tag{10}$$

The objective function 1 maximizes the total collected score. Constraints 2 guarantee that M routes start from location 1 and end at location N. Constraints 3 and 4 determine the connectivity and time line of each tour. Constraints 5 ensure that every location is visited at most once. Constraints 6 and 7 restrict the start of the service to the time window. Constraints 8 limit the time budget. Constraints 9 avoid the violation of the K attribute constraints.

4. Solution algorithm

The algorithm we propose to tackle the MCTOPTW is based on the algorithm proposed by Vansteenwegen et al. [11] for the TOPTW. We have extended it and validated the necessary changes to fit the MCTOPTW. In this section, we start by giving a general description of the TOPTW algorithm and then we focus on the modifications.

4.1. TOPTW algorithm

The heuristic is based on Iterated Local Search (ILS) [17]. Next to the implementation of Vansteenwegen et al. [11], ILS has been successfully applied before to another vehicle routing problem with time windows [18] and other optimization problems such as strip packing [19]. ILS is a metaheuristic method based on iteratively building sequences of solutions generated by an embedded heuristic called local search. This leads to much better solutions than repeating random trials of the same heuristic. The solution found by the local search is perturbed to create a new solution. Then, the best solution is taken as the new starting solution for the local search. The process is repeated until a termination criterion is met. The ILS metaheuristic can be summarized as in Algorithm 1.

Algorithm 1: Diagram of Iterated Local Search

s_0 = GenerateInitialSolution;
s^* = LocalSearch(s_0);
while *termination condition NOT met* **do**
    s' = Perturbation(s^*);
    $s^{*'}$ = LocalSearch(s');
    s^* = AcceptanceCriterion(s^*, $s^{*'}$);

The implementation of the local search heuristic is based on an Insert Step that tries to add new locations to a route, one by one, using two main concepts [11]. The first one is $Wait$, the time a tourist has to wait for a location to be opened. The second one is $MaxShift$, which represents the maximum delay in the arrival to a location without causing a route alteration.

The Insert Step starts with determining the smallest insertion time ($shift_i$) for each location i that can be inserted. For each of these locations a ratio, which weighs the score of the location and the insertion time, is calculated. Among them, the one with the highest ratio is selected for insertion. Then, the Insert Step is repeated until no more locations can be inserted. After an insertion, all other visits to locations are updated.

The perturbation phase is based on a shake movement that removes consecutive visits to locations from a tour. After the removal, all visits following the removed visits are shifted towards the beginning of the route as much as possible, in order to avoid unnecessary waiting.

The heuristic always continues the search from the perturbed solution, it never returns to the best found solution to continue. Of course, the best found solution is always kept in memory. Once the termination criterion is met (maximum number of iterations without improvement or maximum allowed time), the system returns the best solution found. Full details about the TOPTW heuristic are available at Vansteenwegen et al. [11].

4.2. MCTOPTW algorithm

We have adapted this heuristic, which provides very good solutions for the TOPTW, to solve the MCTOPTW. First we have changed the feasibility check of insertions. For the TOPTW only the time feasibility was checked. Inserting extra attribute constraints requires checking each of them to assert the feasibility of the insertion. As the time check is computationally more expensive, for each non included location, we first inspect each constraint feasibility.

The second necessary change is related to the ratio function comparing each point that can be inserted. For the TOPTW, the ratio only takes into account the score of the location and the time required to visit it. The location with the highest ratio according to formula 11 is chosen for insertion.

$$ratio_i = \frac{S_i^2}{shift_i} \tag{11}$$

The quality of the results produced by this ratio is not satisfactory for the MCTOPTW, since the attribute constraints are not taken into account.

Preliminary tests have shown that the best results are obtained by keeping the score at a power of two, instead of increasing the power with the number of attribute constraints. Increasing the power would give an unjustified weight to the score, compared to the sum of the insertion time and the consumption of the attribute constraints.

Regarding the denominator, we analyzed different possibilities in order to define the best ratio function for the MCTOPTW:

- The same weight for all constraints:

$$ratio = \frac{S_i^2}{shift_i + \sum_{k=1}^{K} e_{ik}} \tag{12}$$

- A special weight, $constraintWeight_k$ ($\in [0, 1]$), for each attribute constraint k:

$$ratio = \frac{S_i^2}{shift_i + \sum_{k=1}^{K} constraintWeight_k * e_{ik}} \tag{13}$$

- Include the upper bound for each attribute constraint:

$$ratio = \frac{S_i^2}{\frac{shift_i}{T_{max}} + \sum_{k=1}^{K} \frac{e_{ik}}{E_k}} \tag{14}$$

- Include the quantity of each constraint that is still available in the current solution:

$$ratio = \frac{S_i^2}{\frac{shift_i}{availableTime} + \sum_{k=1}^{K} \frac{e_{ik}}{available_k}} \tag{15}$$

Empirical tests indicated that the best option was a combination of two approaches: give a special weight to each attribute constraint and include the available quantity of each constraint on the route. These tests showed that the optimal weight for the attribute constraints was obtained by setting the weight of each constraint as the inverse of the number of constraints (e.g. 0.5 for two attribute constraints). Thus, we formulate the following formula as the new ratio function:

$$ratio = \frac{S_i^2}{\frac{shift_i}{availableTime} + \sum_{k=1}^{l} \frac{1}{K} \frac{e_{ik}}{available_k}} \tag{16}$$

This ratio function can be motivated intuitively as well. It is not surprising that the quantity that is still available for each constraint is important in this ratio and that this is more relevant than the upper bound itself of a constraint. In this way, the consumption of the most binding constraint will become much more important. Furthermore, with this weighting of the attribute constraints, the insertion time is equally important as the attribute constraints together and more attribute constraints will not increase the total weight of the denominator.

The third change is actually an improvement of the TOPTW algorithm. In the TOPTW, it is possible for locations removed during one iteration to be inserted again immediately during the next iteration. These locations appear to be valuable to obtain a high score solution. Therefore, we try to avoid removing the same locations during consecutive iterations. For each route, a tabu list stores the locations removed during the last two perturbation phases. When a location that appears on the tabu list is considered for removal, the algorithm disregards it and considers the next location for removal. The number of times another location is considered for removal, is dynamically limited to the number of locations in the tabu list. After this amount of tries, the last considered location is removed in any case.

Furthermore, we have made the parameter controlling the maximum number of iterations without improvements, problem dependent. More specifically, the size of the first route of the initial solution is an indication of the number of locations that can be visited per route and, therefore, an indication of the degree of difficulty of the problem. Based on this observation, we decided to fix the maximum number of iterations without improvement ($MaxIter$) according to the size of the first route:

$$MaxIter = FactorNoImprovement * SizeOfFirstRoute \tag{17}$$

Apart from the strength of the perturbation phase (already applied for the TOPTW) and the weight factors of the ratio, we inserted two more parameters to be predefined in this heuristic. The first one is the factor applied to the number of times no improvement is identified ($FactorNoImprovement = 10$). The second one is the number of iterations the removed locations are stored in the tabu list ($TabuIterations = 2$). The actual length of the tabu list will be equal to the number of locations that were removed during these iterations.

Preliminary tests showed that changing the above mentioned parameters did not significantly improve the results and only caused longer computation times. Moreover, small changes in parameter settings appear to have no significant impact on the efficiency of the algorithm. The heuristic we have used to solve the MCTOPTW is shown as a summary in Algorithm 2.

5. Experimental results

We have tested the MCTOPTW heuristic using both existing test problems for the SVRPTW and a new test set. We have designed a new test set based on the available test sets for the

Algorithm 2: MCTOPTW algorithm

```
PositionStartRemove=1;
NumberToRemove=1;
NumberOfTimesNoImprovement=0;
MaxNumberToRemove = N /(3 * M);
MaxIter = FactorNoImprovement * SizeOfFirstRoute;
while NumberOfTimesNoImprovement ¡ MaxIter do
    while not local optimum do
        Insert;
    if Solution better than BestFound then
        BestFound = Solution;
        NumberToRemove = 1;
        NumberOfTimesNoImprovement = 0;
    else
        NumberOfTimesNoImprovement+1;
    Shake Solution (NumberToRemove, PositionStartRemove);
    PositionStartRemove = PositionStartRemove + NumberToRemove;
    NumberToRemove+1;
    PositionStartRemove = PositionStartRemove mod Size of smallest Route;
    if NumberToRemove== MaxNumberToRemove then
        NumberToRemove = 1;
return BestFound
```

TOPTW. We have carried out all computations on a personal computer Intel Core 2 Quad with 2.40 GHz processors and 2 GB Ram. The algorithm is coded using Java 1.6. All results about CPU computation times are shown in seconds.

5.1. SVRPTW

Boussier et al. [16] presented the results of a branch & price based algorithm to solve the SVRPTW. We used these results to test the algorithm presented in this paper. We used Gueguen [14] data sets, which extend data sets defined for the VRPTW [20], for instances with 50 and 100 locations; distance constraints (L) of 50, 100 and 150; and up to 10 vehicles (equals M routes). The algorithm presented in previous sections requires two small modifications so that it can deal with the SVRPTW and can take the limit on the total travel distance into account.

Firstly, the feasibility check of the insertion of location i between locations h and j has to include the covered distance with this simple formula:

$$c_{hi} + c_{ij} - c_{hj} \leq availableDistance \tag{18}$$

Secondly, the ratio in formula 16 has to be calculated as:

$$ratio = \frac{S_i^2}{\frac{shift_i}{availableTime} + \frac{demand_i}{availableCapacity} + \frac{c_{hi}+c_{ij}-c_{hj}}{availableDistance}} \tag{19}$$

Table 1 compares the results of the exact integer solution of Boussier et al. [16] and the heuristic presented in this paper. We present the gap between both solutions and the computation

time of the heuristic in seconds. Besides, the last three rows show the average and the worst gaps and the number of times the heuristic obtains the optimal result.

The computation time is below 5 seconds for most of the problems. The total time for the whole test set is below 150 seconds, while it took around two hours for the exact algorithm of Boussier et al. [16].

The average gap over all SVRPTW instances is 4.4%. However, the gap for two of the smallest problems is above 10%. Without these smallest problems, the average gap would be 3.4% and the worst gap only 9.2%. The optimal routes for the easiest problems ($M \leq 2$; $L = 100$) only contain four or five locations. It appears that our algorithm cannot guarantee a high quality solution for instances where only a few locations can be visited.

However, adding an "insert tabu list", avoiding the insertion of recently removed locations, leads to near optimal results for the smallest problems. For example, with an insert tabu list avoiding to insert the locations removed during the last two iterations, our algorithm obtains optimal results for problems with 50 and 100 locations, distance constraints of 50 and up to 2 routes. On the contrary, this same insert tabu list increases the average gap over all problems to 5.5%, raising the gap for most of the problems above 10%. Furthermore, our practical application of the personalized tourist guide will only have to deal with larger problems. Therefore, the insert tabu list has not been included in our algorithm.

Without the "remove tabu list", the quality of the results would decrease significantly, leading to an average gap of 10.5% (compared to 4.4%). This clearly illustrates the effectiveness of the tabu list. Furthermore, the results show that with the test instances of Boussier et al. [16], the capacity constraint is never binding. If this constraint is removed, the same results can be obtained for all test problems. Thus, although the results for these selective vehicle routing problems with time windows are promising, another set of test problems is required to test the behaviour of the heuristic with multiple attribute constraints.

5.2. New test set

Since no test problems for the MCTOPTW are available in the literature and our algorithm is modified specifically to deal with this variant, we designed a new test set based on the available test sets for the TOPTW. We explain the design of the new test set in detail, to allow other researchers to use the same test set. Righini and Salani [21] created 29 problems for the OPTW using the data set of Solomon [20] for VRPTW and 10 problems using the multi-depot vehicle routing problems of Cordeau et al. [22].

We added attribute constraints to these test problems to design a test set for MCTOPTW. In the available test sets each location has an index. The first attribute's value e_{i1} that is added to each location, is set equal to this index. The second attribute's value e_{i2} is also defined based on this index. The value e_{i2} of the attribute of the first five locations is set equal to 5, the next five locations get the value of 10, the next five locations the value of 15 and the next five locations again get a value of 5, 10, 15, and so on. Table 2 gives an example of the initialization of these attributes for Solomon's problem "c101". Columns x_i and y_i represent the (x, y) coordinates of location i.

It is important to notice that in the data sets the time windows (O_i, C_i) are defined only as bounds for the start of the visit (s_i). As a consequence, the end of the visit can take place after the time window has closed. Righini and Salani [21] provided us with complete details about their optimal OPTW solutions. The maximum value for E_1 and E_2 of each optimal solution has been calculated based on the values associated to the visited locations. We assigned these

values as maximum values for the attribute constraints in the MCOPTW. Ideally, our algorithm should determine for each MCOPTW a solution with a score equal to the optimal score of the corresponding OPTW. We could not access the complete details of solutions from Montemanni and Gambardella [12] and Tricoire et al. [10]. Thus, we could not use their results to create test instances.

We used the above-mentioned problems with the number of routes equal to one and two. To be consistent with the existing literature, we have rounded down the distance calculations for problems from Solomon [20] to one decimal and for problems from Cordeau et al. [22] to two decimals.

Tables 3 and 4 compare the scores obtained by the ILS heuristic for the MCOPTW (instances with one route and two attribute constraints E_1 and E_2) with the optimal results. The first group of columns gives the instance's name, the available time budget, the maximum values associated to the optimal solution for each attribute constraint, and the optimal score. The second group of columns presents the results for problems with one attribute constraint: the score obtained by our algorithm (ILS), the number of locations of the solution, the gap (in %) between the results of our algorithm and the optimal ones, and the computation time (in seconds). The third group of columns presents the results for the same problems with two attribute constraints. For each type of problem, we also show the average and worst gap (in %) and computation time (in seconds). Next to them, we indicate the number of times the heuristic equals the optimal result. The average gap with the optimal results for all tests problems (1 and 2 attribute constraints and 1 route) is 3.9%. Optimal results are obtained for 17 of these 78 test problems.

Tables 5 and 6 compare the scores obtained by the proposed heuristic for the MCTOPTW (instances with more routes) with the best known results for the TOPTW, adding two attribute constraints, E_1 and E_2. For the TOPTW, no optimal solutions are available in the literature. Therefore, the best-known solutions have been used to compare with. These best-known results are obtained by solving the TOPTW with the ILS heuristic without considering any attribute constraints. The attribute constraint values E_1 and E_2 for each instance, have been calculated as the maximum values of the attribute constraints of these TOPTW solutions. With these constraint values, the TOPTW results can be considered also as the best-known results for the MCTOPTW and can be used as benchmarks. Ideally, the proposed heuristic should determine for each MCTOPTW a solution with a score equal or better than the best-known score of the corresponding TOPTW.

The first group of columns of tables 5 and 6 gives the instance's name, the available time budget, the value of the attribute constraints associated to the best-known solution and the score of the best-known solution. The second group of columns presents the results for problems with one attribute constraint: the score obtained by the heuristic, the gap (in %) between the results of our algorithm and the best-known solution, and the computation time in seconds. The third group of columns presents the results for the same problems adding a second attribute constraint. For each type of problem the average and worst gap (in %) and computation time (in seconds) are shown. Next to them, the number of times the heuristic equals or improves the best-known result is indicated. The average gap with the best known results for all tests problems (1 and 2 attribute constraints and 2 routes) is 0.8%. The worst gap is only 5.7% and the best-known results are obtained for half of these 78 test problems.

The results for all new test instances are summarized in Table 7. We have divided the table in two rows, one for problems with one route and the other for problems with two routes. Each of them is again divided according to the kind of MC(T)OPTW problems: Solomon [20] (with 100 locations), and pr (Cordeau et al. [22], with 44-288 locations).

We have grouped the columns according to the number of attribute constraints used, one or two. The table shows the following data for each case:

- The average and worst gap between the results obtained by our algorithm and the optimal (1 route) or best-known (2 routes) solutions.
- The average and worst computation time in seconds required to solve the MC(T)OPTW instances.
- The number of times our algorithm equals the optimal result, or equals or improves the best known solution.

The average gap for all 156 instances (up to 288 locations with one and two attribute constraints, and one and two routes) is below 2.5% and the average computation time is below 3 seconds. Although the worst gap for problems with one route is 12.7%, the gap is always below 10%, except for 3 problems. For problems with one route the heuristic is able to find the optimal solutions in 22% of the problems. For problems with two routes, in 40% of the problems the best-known solution is obtained, the average gap over all instances is 1% and the average computation time is around 4.4 seconds. Based on these results, it can be concluded that the quality and the computation times of the results are independent of the number of attribute constraints. Furthermore, high quality results are obtained in only a few seconds of computation time, also for very difficult instances.

6. Conclusions and further work

This paper shows how tourist routes matching a tourist's profile can be generated by solving a Multi Constrained Team Orienteering Problem with Time Windows (MCTOPTW). It introduces an Iterated Local Search (ILS) based algorithm to solve the MCTOPTW fast and effectively. The algorithm performs well on a published set of Selective Vehicle Routing Problems with Time Windows (SVRPTW) and a large set of new MCTOPTW instances. The algorithm has been developed to be applied inside a Personalized Electronic Tourist Guide (PET), a mobile handheld device creating tourist routes maximizing tourists' satisfaction.

The metaheuristic is based on an existing metaheuristic for the TOPTW. We have adapted and extended it to solve the MCTOPTW. We have included new feasibility checks, a new ratio function to compare possible insertions, and a tabu list inside the perturbation phase. The tabu list, avoiding to remove locations that were recently removed, improves the quality of the results significantly, with small computational cost. Finally, we have validated the quality of the algorithm. We have compared our results with existing results for the SVRPTW, and we have proposed new test sets to allow researchers to compare new heuristics with our solutions.

Compared to an exact method published for the SVRPTW, our algorithm proves to solve the problem efficiently, with an overall average gap of 4.4% and an overall average computation time of only 2.4 seconds. Only for instances where only four or five locations can be visited, our algorithm cannot guarantee a high quality solution. Nevertheless, the algorithm can be modified easily to deal with these small instances efficiently. The total time for the whole test set is below 150 seconds, while it took around two hours for the exact method.

Based on the OPTW test sets, we developed new test instances for the MCTOPTW with one and two routes and one and two attribute constraints. For 39 problems with one route and two attribute constraints, the average gap with the optimal results is only 3.9%. The average

computation time is 1.1 seconds. For problems with two routes, the average gap with the best-known results is 0.9% and the average computation time is 4 seconds. The performance of the ILS heuristic appears to be independent of the number of attribute constraints. Due to its simplicity and the high quality results, the algorithm can easily be applied for problems with more attribute constraints.

To the best of our knowledge, this is the first heuristic to specifically tackle MCTOPTW problems. Furthermore, the algorithm also performs very well in dealing with SVRPTW instances. The new test instances and our results can be used as a benchmark for further research. In order to fully test the capacity of the heuristic to solve the MCTOPTW, optimal solutions for the (MC)TOPTW should be calculated and published.

For the Personalized Electronic Tourist guide and other tourist applications, the algorithm is suitable to tackle Tourist Trip Design Problems. An important extension of the current problem is to take into account different means of (public) transportation. Therefore, the arcs between locations should model different transportation means. One could be the walking distance, based on the road network. However, for public transportation, an independent network of transport stops should be included in the problem.

Acknowledgments

The authors would like to thank the Basque Government for partially funding this work through the neurebide and etourgune projects and to the Centre for Industrial Management of the Katholieke Universiteit Leuven for hosting Ander Garcia as a guest researcher during 2008. Pieter Vansteenwegen is a post-doctoral research fellow of the "Fonds Wetenschappelijk Onderzoek - Vlaanderen (FWO)". Finally, authors would like to thank Righini and Salani for providing detailed solutions.

References

References

[1] P. Vansteenwegen, Planning in tourism and public transportation, Ph.D. thesis, Centre for Industrial Management, Katholieke Universiteit Leuven, Belgium, 2008.

[2] W. Souffriau, P. Vansteenwegen, J. Vertommen, G. Vanden Berghe, D. Van Oudheusden, A Personalized Tourist Trip Design Algorithm for Mobile Tourist Guides, Appl. Articial Intelligence 22 (10) (2008) 964–985.

[3] A. Garcia, M. Linaza, O. Arbelaitz, P. Vansteenwegen, Intelligent Routing System for a Personalised Electronic Tourist Guide, in: Inform. and Comm. Technologies in Tourism 2009, SpringerWienNewYork, Amsterdam, The Netherlands, ISBN 978-3-211-93970-3, 185–197, 2009.

[4] P. Vansteenwegen, W. Souffriau, G. Vanden Berghe, D. Van Oudheusden, Metaheuristics for tourist trip planning, in: M. Geiger, W. Habenicht, M. Sevaux, K. Sorensen (Eds.), Metaheuristics in the Service Industry. Lecture Notes in Economics and Mathematical Systems, Springer Verlag, 15–31, 2009.

[5] B. Zenker, B. Ludwig, ROSE Assisting Pedestrians to Find Preferred Events and Comfortable Public Transport Connections, in: ACM Mobility Conference, Nice, France, 2009.

[6] T. Tsiligirides, Heuristic methods applied to orienteering, J. Oper. Res. Soc. 35 (9) (1984) 797–809.

[7] P. Vansteenwegen, W. Souffriau, D. Van Oudheusden, The orienteering problem: a survey, Eur. J. Oper. Res.: under review .

[8] M. W. P. Savelsbergh, Local search in routing problems with time windows, Ann. Oper. Res. 4 (1) (1985) 285–305.

[9] G. Righini, M. Salani, Decremental state space relaxation strategies and initialization heuristics for solving the Orienteering Problem with Time Windows with dynamic programming, Comput. Oper. Res. 36 (4) (2009) 1191–1203, ISSN 0305-0548.

[10] F. Tricoire, M. Romauch, K. Doerner, R. Hartl, Heuristics for the multi-period orienteering problem with multiple time windows, Comput. Oper. Res. 37 (2) (2010) 351–367.

[11] P. Vansteenwegen, W. Souffriau, G. Vanden Berghe, D. Van Oudheusden, Iterated local search for the team orienteering problem with time windows, Comput. Oper. Res. 36 (12) (2009) 3281–3290.
[12] R. Montemanni, L. Gambardella, Ant Colony System for Team Orienteering Problems with Time Windows, Found. Comput. Decision Sci. 34 (4) (2009).
[13] C. Archetti, D. Feillet, A. Hertz, M. Speranza, The Capacitated Team Orienteering and Profitable Tour Problems, Tech. Rep. G200731, Les Cahiers du GERAD, Canada, 2007.
[14] C. Gueguen, Methodes de resolution exacte pour les problemes de tournes de vehicules, Ph.D. thesis, Ecole Centrale Paris, France, 1999.
[15] N. Hayari, M. A. Manier, C. Bloch, A. El Moudni, Un algorithme evolutionniste pour le probleme de tournees selectives avec contraintes de fenetres de temps (in French), in: 4e Conf. Francophone de MOdelisation et SIMulation, Toulouse, France, 2003.
[16] S. Boussier, D. Feillet, M. Gendreau, An exact algorithm for team orienteering problems, 4OR-Q. J. Oper. Res. 5 (3) (2006) 211–230.
[17] H. R. Lourenço, O. Martin, T. Stutzle, Iterated Local Search, in: F. G. Kochenberger, G. (Eds.), Handbook of Metaheuristics, Kluwer Academic Publishers, 321–353, 2003.
[18] H. Hashimoto, M. Yagiura, T. Ibaraki, An iterated local search algorithm for the time-dependent vehicle routing problem with time windows, Discrete Optim. 5 (2008) 434–456.
[19] T. Imamichi, M. Yagiura, H. Nagamochi, An iterated local search algorithm based on nonlinear programming for the irregular strip packing problem, Discrete Optim. 6 (2009) 345–361.
[20] M. M. Solomon, Algorithms for the vehicle routing and scheduling problems with time window constraints, Oper. Res. 35 (2) (1987) 254–265, ISSN 0030-364X.
[21] G. Righini, M. Salani, New dynamic programming algorithms for the resource constrained elementary shortest path problem, Networks 51 (3) (2008) 155–170.
[22] J. Cordeau, M. Gendreau, G. Laporte, A tabu search heuristic for periodic and multi-depot vehicle routing problems, Networks 30 (2) (1997) 105–119.

Table 1: Results compared to SVRPTW

L	M	50 locations				100 locations			
		Boussier	ILS	GAP	CPU(s)	Boussier	ILS	GAP	CPU(s)
50	1	123	123	0,0%	0,3	164	121	26,2%	0,2
	2	215	215	0,0%	0,6	313	205	34,5%	0,3
	3	282	277	1,8%	0,4	445	432	2,9%	0,8
	4	329	317	3,6%	0,3	574	561	2,3%	1,1
	5	369	369	0,0%	0,5	693	661	4,6%	1,5
	6	407	398	2,2%	0,5	767	767	0,0%	4,0
	7	439	430	2,1%	0,6	851	795	6,6%	1,8
	8	456	447	2,0%	0,6	925	886	4,2%	1,7
	9	456	456	0,0%	0,7	995	926	6,9%	2,2
	10	456	456	0,0%	0,7	1042	946	9,2%	2,2
100	1	202	202	0,0%	0,3	280	271	3,2%	0,5
	2	374	350	6,4%	0,5	541	524	3,1%	1,0
	3	520	515	1,0%	0,8	770	735	4,5%	2,1
	4	651	619	4,9%	1,1	983	930	5,4%	1,9
	5	761	743	2,4%	1,4	1190	1132	4,9%	2,9
	6	852	836	1,9%	1,7	1372	1326	3,4%	5,2
	7	935	909	2,8%	1,8	1538	1447	5,9%	4,9
	8	1006	979	2,7%	6,4	1675	1620	3,3%	7,8
	9	1071	1004	6,3%	1,6	1814	1701	6,2%	4,5
	10	1125	1059	5,9%	1,7	1930	1835	4,9%	8,8
150	1	210	210	0,0%	0,3	320	309	3,4%	0,6
	2	383	369	3,7%	0,5	590	562	4,7%	1,1
	3	542	526	3,0%	0,8	820	787	4,0%	2,5
	4	695	694	0,1%	2,1	1049	1008	3,9%	2,7
	5	819	774	5,5%	1,5	1255	1238	1,4%	7,0
	6	917	890	2,9%	1,4	1444	1402	2,9%	6,3
	7	1003	967	3,6%	2,2	1606	1547	3,7%	5,7
	8	1068	1033	3,3%	1,9	1752	1673	4,5%	10,8
	9	1116	1028	7,9%	1,4	1894	1779	6,1%	5,8
	10	1146	1096	4,4%	1,9	2012	1944	3,4%	8,3
			average	2,7%	1,2		average	6,0%	3,5
			worst	7,9%	6,4		worst	34,5%	10,8
			#optimal	7			#optimal	1	

Table 2: Initialization of parameters for MCTOPTW

i	x_i	y_i	T_i	S_i	O_i	C_i	e_{i1}	e_{i2}	i	x_i	y_i	T_i	S_i	O_i	C_i	e_{i1}	e_{i2}
0	40	50			0	1236			10	35	66	90	10	357	410	10	10
1	45	68	90	10	912	967	1	5	11	35	69	90	10	448	505	11	15
2	45	70	90	30	825	870	2	5	12	25	85	90	20	652	721	12	15
3	42	66	90	10	65	146	3	5	13	22	75	90	30	30	92	13	15
4	42	68	90	10	727	782	4	5	14	22	85	90	10	567	620	14	15
5	42	65	90	10	15	67	5	5	15	20	80	90	40	384	429	15	15
6	40	69	90	20	621	702	6	10	16	20	85	90	40	475	528	16	5
7	40	66	90	20	170	225	7	10	17	18	75	90	20	99	148	17	5
8	38	68	90	20	255	324	8	10	18	15	75	90	20	179	254	18	5
9	38	70	90	10	534	605	9	10	19	15	80	90	10	278	345	19	5
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

Table 3: Results for Solomon's test problems with 100 locations, 1 and 2 attribute constraints and 1 route

Name	T_{max}	E_1	E_2	*Optimal*	1 constraint				2 constraints			
					ILS	#	*GAP*	*CPU*(*s*)	*ILS*	#	*GAP*	*CPU*(*s*)
c101	1236	556	100	320	300	10	6,3%	0,3	320	10	0,0%	0,2
c102	1236	801	100	360	360	11	0,0%	0,4	360	11	0,0%	0,3
c103	1236	680	100	400	380	10	5,0%	0,3	390	10	2,5%	0,4
c104	1236	514	105	420	400	10	4,8%	0,3	400	10	4,8%	0,3
c105	1236	636	100	340	330	10	2,9%	0,3	340	10	0,0%	0,2
c106	1236	565	105	340	340	10	0,0%	0,3	340	10	0,0%	0,2
c107	1236	405	115	370	370	11	0,0%	0,5	340	11	8,1%	0,2
c108	1236	405	115	370	350	11	5,4%	0,3	340	11	8,1%	0,2
c109	1236	534	105	380	380	11	0,0%	0,3	370	10	2,6%	0,3
r101	230	434	105	198	182	7	8,1%	0,1	182	7	8,1%	0,1
r102	230	684	90	286	281	10	1,7%	0,3	286	11	0,0%	0,2
r103	230	659	95	293	286	10	2,4%	0,4	286	10	2,4%	0,2
r104	230	613	90	303	288	10	5,0%	0,3	288	10	5,0%	0,2
r105	230	711	100	247	247	11	0,0%	0,3	247	11	0,0%	0,2
r106	230	645	95	293	281	10	4,1%	0,3	289	11	1,4%	0,2
r107	230	733	140	299	289	11	3,3%	0,4	288	10	3,7%	0,2
r108	230	765	140	308	308	13	0,0%	0,7	295	11	4,2%	0,3
r109	230	760	120	277	276	11	0,4%	0,3	277	12	0,0%	0,3
r110	230	790	135	284	274	11	3,5%	0,6	277	12	2,5%	0,3
r111	230	721	110	297	295	11	0,7%	0,5	295	11	0,7%	0,3
r112	230	752	115	298	292	11	2,0%	0,4	295	11	1,0%	0,3
rc101	240	420	75	219	216	9	1,4%	0,2	219	10	0,0%	0,2
rc102	240	333	80	266	259	9	2,6%	0,2	259	9	2,6%	0,1
rc103	240	333	80	266	259	9	2,6%	0,2	259	9	2,6%	0,1
rc104	240	347	100	301	301	11	0,0%	0,2	296	11	1,7%	0,1
rc105	240	399	90	244	213	8	12,7%	0,2	213	8	12,7%	0,1
rc106	240	512	95	252	233	9	7,5%	0,5	227	9	9,9%	0,1
rc107	240	489	100	277	270	10	2,5%	0,3	270	10	2,5%	0,2
rc108	240	406	100	298	298	11	0,0%	0,4	298	11	0,0%	0,2
					average		2,9%	0,3	average		3,0%	0,2
					max		12,7%	0,7	max		12,7%	0,4
					#			8	#			9

Table 4: Results for test problems of Cordeau, Gendreau and Laporte , 1 and 2 attribute constraints and 1 route

Name	T_{max}	E_1	E_2	$Optimal$	1 constraint				2 constraints			
					ILS	#	GAP	$CPU(s)$	ILS	#	GAP	$CPU(s)$
pr01	1000	507	195	308	290	17	5,8%	0,4	288	19	6,5%	0,5
pr02	1000	1190	230	404	375	18	7,2%	1,0	378	19	6,4%	2,0
pr03	1000	1309	180	394	380	20	3,6%	1,7	386	21	2,0%	1,6
pr04	1000	2076	220	489	445	22	9,0%	2,5	455	24	7,0%	3,2
pr05	1000	3562	300	595	521	26	12,4%	10,0	553	28	7,1%	8,4
pr06	1000	4284	260	565	534	26	5,5%	10,4	493	22	12,7%	6,0
pr07	1000	585	165	298	289	15	3,0%	0,5	289	15	3,0%	0,5
pr08	1000	1800	215	463	452	24	2,4%	3,9	450	23	2,8%	3,2
pr09	1000	3562	300	493	461	26	6,5%	3,7	449	23	8,9%	5,4
pr10	1000	4352	235	552	517	26	6,3%	9,0	502	25	9,1%	7,2
					average		6,2%	4,3	average		6,6%	3,8
					max		12,4%	10,4	max		12,7%	8,4
					#			0	#			0

Table 5: Results for Solomon's test problems with 100 locations, 1 and 2 attribute constraints and 2 routes

Name	T_{max}	E_1	E_2	*Best*	1 constraint			2 constraints		
					ILS	*GAP*	$CPU(s)$	*ILS*	*GAP*	$CPU(s)$
c101	1236	614	100	590	580	1,7%	1,2	580	1,7%	1,0
c102	1236	801	100	650	650	0,0%	1,3	650	0,0%	1,0
c103	1236	727	105	710	710	0,0%	2,2	690	2,8%	1,2
c104	1236	624	110	760	760	0,0%	1,2	740	2,6%	1,0
c105	1236	656	105	640	640	0,0%	1,2	640	0,0%	1,2
c106	1236	604	100	620	620	0,0%	1,1	620	0,0%	0,9
c107	1236	648	90	670	660	1,5%	1,3	670	0,0%	1,4
c108	1236	641	110	680	680	0,0%	1,3	680	0,0%	1,2
c109	1236	788	100	710	710	0,0%	1,5	720	-1,4%	1,2
r101	230	380	70	341	322	5,6%	0,3	322	5,6%	0,3
r102	230	684	90	501	508	-1,4%	1,3	494	1,4%	0,9
r103	240	566	130	513	512	0,2%	1,4	513	0,0%	1,1
r104	240	639	90	531	538	-1,3%	1,3	518	2,4%	1,0
r105	240	587	75	430	434	-0,9%	0,6	423	1,6%	0,7
r106	240	645	95	529	529	0,0%	1,7	529	0,0%	1,2
r107	240	566	75	527	523	0,8%	0,9	527	0,0%	1,1
r108	240	639	90	534	539	-0,9%	1,0	541	-1,3%	1,1
r109	240	735	105	506	498	1,6%	1,7	488	3,6%	1,0
r110	240	738	125	506	519	-2,6%	1,5	503	0,6%	1,6
r111	1000	737	125	535	536	-0,2%	1,1	530	0,9%	1,9
r112	1000	776	120	522	513	1,7%	1,3	520	0,4%	1,1
rc101	1000	557	60	421	427	-1,4%	1,2	427	-1,4%	0,8
rc102	1000	650	70	487	497	-2,1%	1,8	487	0,0%	1,3
rc103	1000	527	90	512	501	2,1%	0,7	510	0,4%	0,6
rc104	1000	531	120	551	556	-0,9%	1,4	551	0,0%	1,6
rc105	1000	592	90	451	448	0,7%	0,6	448	0,7%	0,6
rc106	1000	600	85	464	462	0,4%	1,0	455	1,9%	1,0
rc107	1000	531	80	520	516	0,8%	1,6	523	-0,6%	0,9
rc108	1000	527	85	540	526	2,6%	1,1	541	-0,2%	0,9
					average	0,3%	1,2	average	0,7%	1,1
					max	5,6%	2,2	max	5,6%	1,9
					#		17	#		15

Table 6: Results for test problems of Cordeau, Gendreau and Laporte with 1 and 2 attribute constraints and 2 routes

Name	T_{max}	E_1	E_2	$Best$	1 constraint			2 constraints		
					ILS	GAP	$CPU(s)$	ILS	GAP	$CPU(s)$
pr01	230	506	292	481	489	-1,7%	0,9	479	0,4%	1,0
pr02	230	1001	366	685	654	4,5%	4,4	656	4,2%	3,2
pr03	230	1380	325	692	701	-1,3%	6,1	710	-2,6%	5,3
pr04	230	2523	435	880	872	0,9%	18,1	846	3,9%	10,5
pr05	230	3925	494	1031	1002	2,8%	42,7	1036	-0,5%	27,5
pr06	230	4198	559	992	952	4,0%	20,9	935	5,7%	20,0
pr07	230	585	298	560	547	2,3%	2,8	546	2,5%	1,9
pr08	230	1904	357	809	774	4,3%	6,5	813	-0,5%	11,4
pr09	230	2967	410	819	828	-1,1%	19,8	823	-0,5%	14,8
pr10	230	4033	529	1037	998	3,8%	27,6	1012	2,4%	28,5
					average	1,9%	15,0	average	1,5%	12,4
					max	4,5%	42,7	max	5,7%	28,5
					#		3	#		4

Table 7: Summary of new test set's result

routes	Type		1 constraint		2 constraints	
			MCOPTW	CPU(s)	MCOPTW	CPU(s)
1	100	average	2,9%	0,3	3,0%	0,2
		worst	12,7%	0,7	12,7%	0,4
		#equal	8		9	
	pr	average	6,2%	4,3	6,6%	3,8
		worst	12,4%	10,4	12,7%	8,4
		#equal	0		0	
2	100	average	0,3%	1,2	0,7%	1,1
		worst	5,6%	2,2	5,6%	1,9
		#equal	17		15	
	pr	average	1,9%	15,0	1,5%	12,4
		worst	4,5%	42,7	5,7%	28,5
		#equal	3		4	